实验编号：2 **四川师大《IOS高级开发技术》实验报告 2018** 年**9**月**12**日

**计算机科学学院**2016 级 4班 实验名称： 作业二 \_

姓名：\_\_邹琳\_\_\_ 学号：\_\_\_2016110458\_\_ 指导老师：\_李贵洋\_ 实验成绩:\_\_\_\_\_

**实验\_二\_ \_\_\_\_\_作业二\_\_\_\_\_**

1. 实验目的与要求

1、实验目的

1. 掌握闭包、扩展、泛型的定义;
2. 掌握排序等采用闭包方法的实现；
3. 掌握类的派生、协议；
4. 掌握版本控制git的进阶使用；

2、实验要求

1. 认真填写实验报告，要求附加部分运行界面和主要代码；
2. 对设计好的程序，检查输出是否符合预期，如有错请分析错误原因并解决；

二. 实验内容

作业1（闭包、扩展、泛型）：

1. 给定一个Dictionary，Dictionary包含key值name和age，用map函数返回age字符串数组;
2. 给定一个String数组，用filter函数选出能被转成Int的字符串
3. 用reduce函数把String数组中元素连接成一个字符串，以逗号分隔
4. 用 reduce 方法一次求出整数数组的最大值、最小值、总数和
5. 新建一个函数数组，函数数组里面保存了不同函数类型的函数，要求从数组里找出参数为一个整数，返回值为一个整数的所有函数；
6. 扩展Int，增加sqrt方法，可以计算Int的Sqrt值并返回浮点数，进行验证；
   1. 实现一个支持泛型的函数，该函数接受任意个变量并返回最大和最小值，分别传入整数值、浮点数值、字符串进行验证。
7. 掌握版本控制git的进阶使用
   1. git reset 恢复到之前修改的版本；
   2. git log 看提交记录；
   3. git branch 新建分支；
   4. git checkout 切换分支；
   5. git branch -d 删除分支；

作业2:（枚举、类、派生、协议）(红色字体为新增内容)

1. 实现Person类：
   1. 要求具有firstName, lastName，age，gender等存储属性,fullName计算属性；其中gender是枚举类型（male，female）；
   2. 具有指定构造函数和便利构造函数；
   3. 两个Person实例对象可以用==和!=进行比较；
   4. Person实例可以直接用print输出；
   5. Person增加run方法(方法里面直接print输出Person XXX is running;
2. 从Person分别派生Teacher类和Student类：
   1. Teacher类增加属性title，实例可以直接用print输出；
   2. Student类增加属性stuNo，实例可以直接用print输出；
   3. Teacher和Student重载run方法(方法里面直接print输出Teacher XXX is running和Student XXX is running)
3. 分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；
4. 新建一个协议SchoolProtocol，协议包括一个department属性(Enum，自己实现enum的定义)和lendBook方法（随便写点内容，能区隔即可）；
5. 修改Teacher和Student，让这两个类实现该协议；
6. 对数组执行以下要求：
   1. 分别统计Person、Teacher和Student对象的个数并放入一字典中，统计完后输出字典内容；
   2. 对数组按以下要求排序并输出：age、fullName、gender+age；
   3. 对数组进行穷举，调用每个对象的run方法，同时调用满足协议SchoolProtocol对象的lendBook方法；

三.实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）

作业1（闭包、扩展、泛型）：

//: Playground - noun: a place where people can play

import UIKit

//第1题

let dictionary:[[String:Any]] = [["age":18,"name":"zhangsan"],["age":20,"name":"lisi"]]

let age = dictionary.map({$0["age"]})

print(age)

//第2题

func filter(str:[String]) -> [String]{

var str2:[String] = [String]()

for str3 in str{

if Int(str3) != nil {

str2.append(str3)

}

}

return str2

}

var str2 = [ "11111","22222","33333","aaaaaa","bbbbb","ccccc"]

let transfer = filter(str:str2)

print(transfer)

//第3题

let array = ["apple","pear","banana","orange"]

let connectarray = array.reduce(""){

connectarray,letter in "\(connectarray),\(letter)"

}

print(connectarray)

//第4题

let arr = [12,24,48,36,96]

let temp = arr.reduce((max:Int.min,min:Int.max,sum:0)){

(temp,i) in return(max(temp.max,i),min(temp.min,i),temp.sum+i)

}

print("max:\(temp.max)")

print("min:\(temp.min)")

print("sum:\(temp.sum)")

//第5题

var arrfunc = [Any]()

func f1(x1: String) -> String {

return x1

}

func f2(x2: Int) -> Int {

return x2

}

func f3(x3: Double, x4:Double) -> Int{

return 4

}

func f4(x5: Int) -> Int{

return x5

}

arrfunc.append(f1)

arrfunc.append(f2)

arrfunc.append(f3)

arrfunc.append(f4)

var j = 1

for i in arrfunc{

if i is ((Int)->Int){

print("func f\(j) is:(Int) -> Int")

}

j = j + 1

}

//第6题

extension Int{

func sqrt(number:Int)->Double{

return Darwin.sqrt(Double(number))

}

}

print(sqrt(16))

//第7题

func MaxandMin<T:Comparable>(data: [T]) -> (max:T,min:T){

var max = data[0]

var min = data[0]

for i in data{

if max < i{

max = i

}else if min > i{

min = i

}

}

return (max,min)

}

let intapp = MaxandMin(data:[10,12,14,16,8,4])

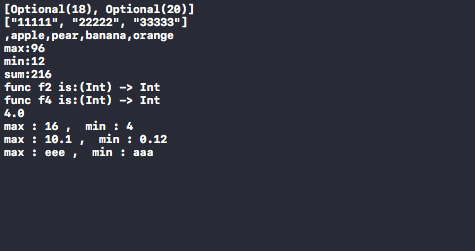
print("max : \(intapp.max) , min : \(intapp.min)")

let doubleapp = MaxandMin(data:[10.1,0.12,1.4,1.65,8.2,4.1])

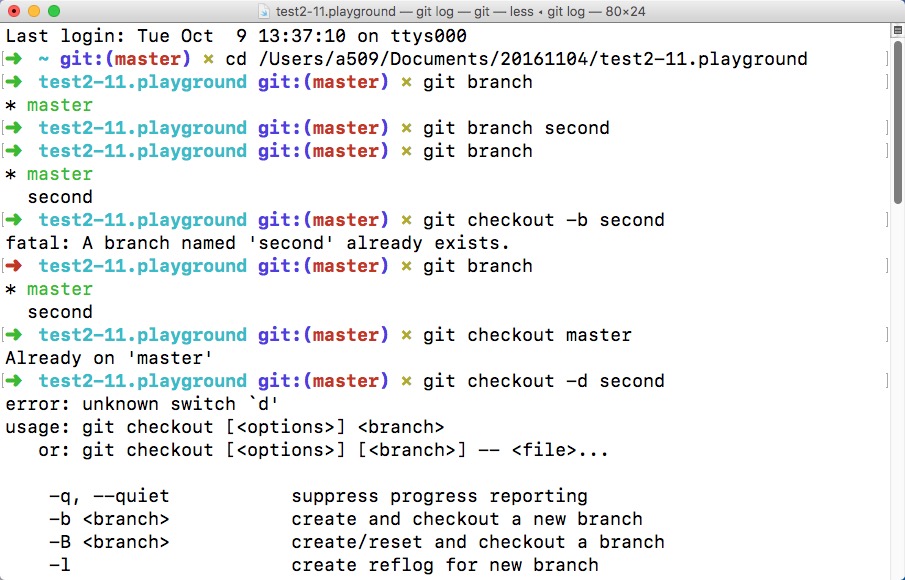
print("max : \(doubleapp.max) , min : \(doubleapp.min)")

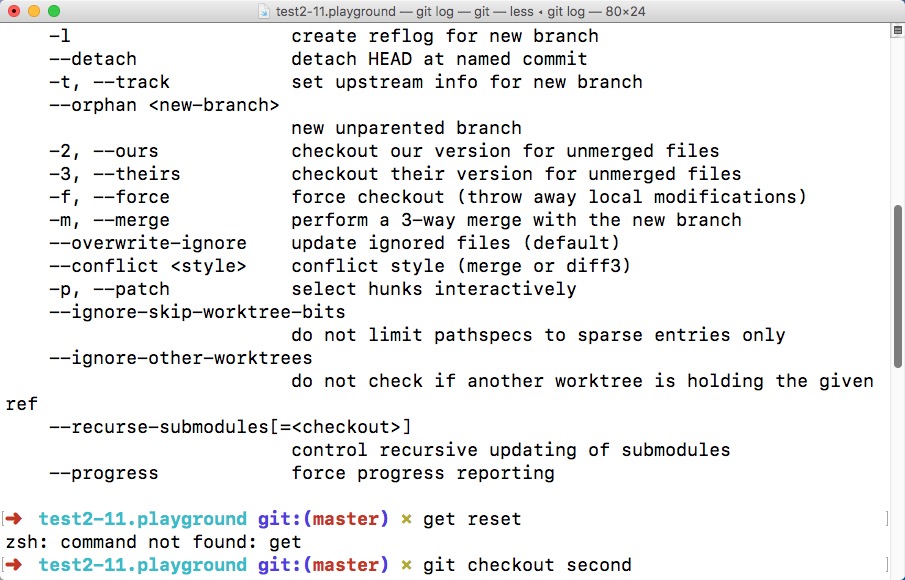
let charapp = MaxandMin(data:["aaa","bbb","ccc","ddd","eee"])

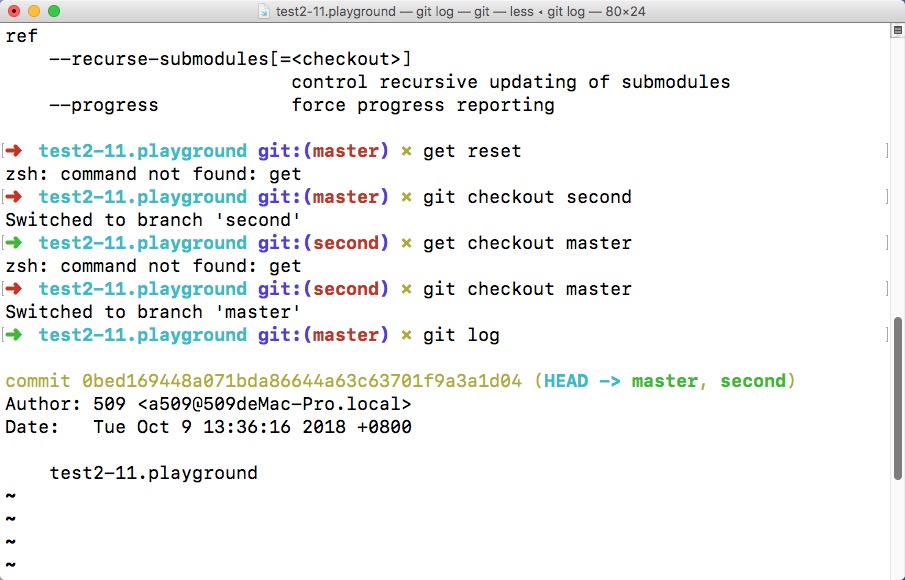
print("max : \(charapp.max) , min : \(charapp.min)")



Git 进阶使用：







作业2:（枚举、类、派生、协议）(红色字体为新增内容)

//: Playground - noun: a place where people can play

import UIKit

var str = "Hello, playground"

enum Gender:Int{

case male

case female

static func >(Ihs:Gender,rhs:Gender)->Bool{

return Ihs.rawValue < rhs.rawValue

}

}

enum Enum{

case math

case chinese

case english

case null

}

//协议

protocol SchoolProtocol {

var des:Enum{get}

mutating func lendBook()

}

class Person{//Person类(父类)

var firstName:String

var lastName:String

var age:Int

var gender:Gender

var fullName:String{

get{

return firstName+""+lastName

}

}

static func == (p1:Person,p2:Person)->Bool {

return p1.firstName == p2.firstName

}

init(firstName:String,lastName:String,age:Int,gender:Gender){//初始构造函数

self.firstName = firstName

self.lastName = lastName

self.age = age

self.gender = gender

}

convenience init(name:String){//便利构造函数

self.init(firstName:name,lastName:"",age:18,gender:Gender.female)

}

var description:String{//实现Person实例可以直接用print输出

return "Name:\(fullName) Age:\(age) Gender:\(gender) "

}

func run(){

print("Person"+fullName+" is running")

}

}

class Student:Person,SchoolProtocol{//由Person派生出Student子类

var stuNo:String

init(firstName:String,lastName:String,age:Int,gender:Gender,stuNo:String){

self.stuNo = stuNo

super.init(firstName:firstName,lastName:lastName,age:age,gender:gender)

//调用父类的构造函数

}

convenience init(name:String){

self.init(firstName:name,lastName:"",age:18,gender:Gender.male,stuNo:"2016110401")

}

override var description:String{

return super.description + " stuNo:\(stuNo)"

}

override func run(){

print("Student"+fullName+" is running")

}

var des: Enum = Enum.chinese

func lendBook() {

print("student lend books")

}

}

class Teacher:Person,SchoolProtocol{ //由Person派生出Teacher子类

var title:String

init(firstName:String,lastName:String,age:Int,gender:Gender,title:String){

self.title = title

super.init(firstName:firstName,lastName:lastName,age:age,gender:gender)

//调用父类的构造函数

}

convenience init(name:String){

self.init(firstName:name,lastName:"",age:36,gender:Gender.male,title:"math")

}

override var description:String{

return super.description + " title:\(title)"

}

override func run(){

print("Teacher"+fullName+" is running")

}

var des: Enum = Enum.math

func lendBook() {

print("teacher lend books")

}

}

//实例化对象并输出

let p1 = Person(firstName:"xiao",lastName:"ming",age:16,gender:Gender.male)

p1.run()

let s1 = Student(firstName:"xiao",lastName:"hong",age:19,gender:Gender.female,stuNo:"201")

s1.run()

s1.lendBook()

let t1 = Teacher(firstName:"xiao",lastName:"fang",age:32,gender:Gender.female,title:"math")

t1.run()

t1.lendBook()

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//分别构造多个Person、Teacher和Student对象，并将这些对象存入同一个数组中；

var people = [Person]()//初始化一个空的Person类型数组

people.append(p1)//将不同对象存入数组

people.append(s1)

//生成3个Person对象

for i in 1...3 {

let p = Person(firstName: "李", lastName: "\(i)", age: 30, gender:Gender.male)

people.append(p)

}

//生成3个Student对象

for i in 1...3 {

let s = Student(firstName:"赵",lastName: "\(i)",age:18,gender:Gender.female,stuNo:"20160100\(i)")

people.append(s)

}

//生成3个Teacher对象

for i in 1...3 {

let t = Teacher(firstName:"王",lastName:"\(i)",age:20,gender:Gender.male,title:"数学")

people.append(t)

}

print(people)

//定义一个字典，用于统计每个类的对象个数

var dictionary = ["Person":0,"Student":0,"Teacher":0]

for item in people {

if item is Student {

dictionary["Teacher"]! += 1

}else if item is Teacher {

dictionary["Student"]! += 1

}else{

dictionary["Person"]! += 1

}

}

for (key,value) in dictionary {

print("\(key) has \(value) items")

}

for item in people{

print(item.description)

}

//按照age值从小到大排序

people.sort{

return $0.age < $1.age

}

for item in people {

print(item.description)

}

//按照fullName从小到大排序

people.sort{

return $0.fullName < $1.fullName

}

for item in people {

print(item.description)

}

//按照gender+age从小到大排序

people.sort{

return ($0.gender > $1.gender) && ($0.age > $1.age)

}

for item in people {

print(item.description)

}

for p in people{

if p is Student {

(p as! Student).lendBook()

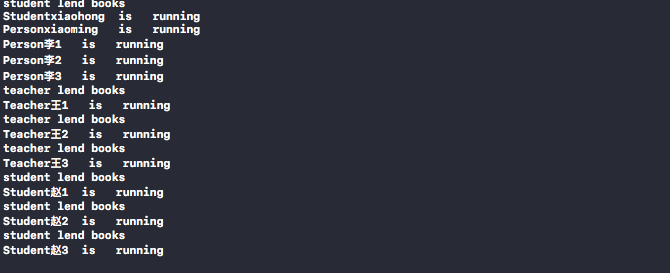
}else if p is Teacher{

(p as! Teacher).lendBook()

}

p.run()

}



四．实验结果的分析与评价（该部分如不够填写，请另加附页）

本次实验主要考察了闭包、扩展、泛型，枚举、类、派生、协议等swift语法实验中要注意闭包的语法规范及协议的定义与使用，注意继承于实现协议的区别，另外掌握了git版本控制的几种基础语法，方便对代码的管理与维护。

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。